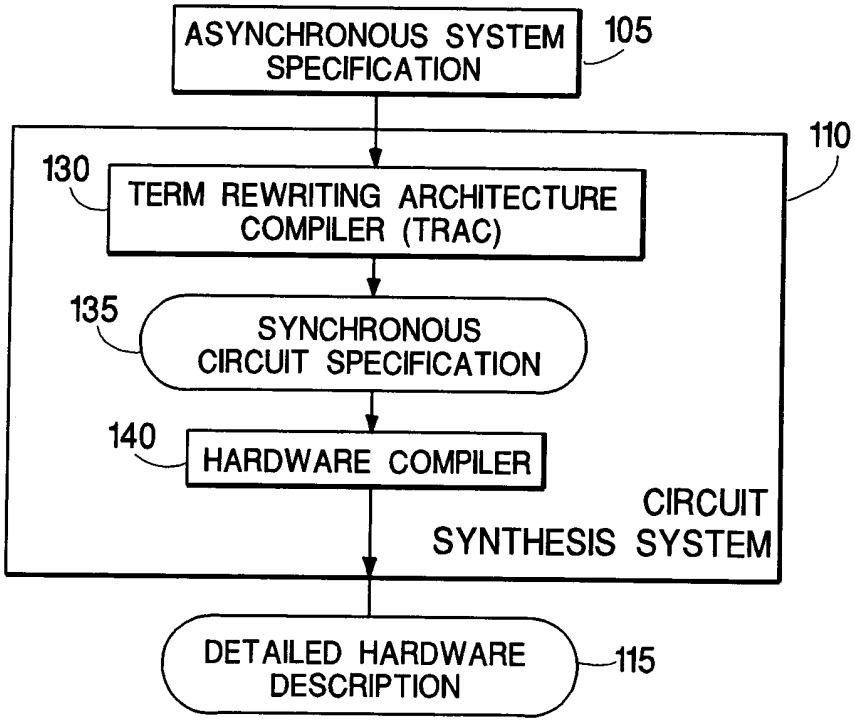
ASYNCHRONOUS SYSTEM SPECIFICATION
105
110
130
TERM REWRITING ARCHITECTURE COMPILER (TRAC)
135
SYNCHRONOUS CIRCUIT SPECIFICATION
140
HARDWARE COMPILER
CIRCUIT SYNTHESIS SYSTEM
DETAILED HARDWARE DESCRIPTION
115

FIG. 1

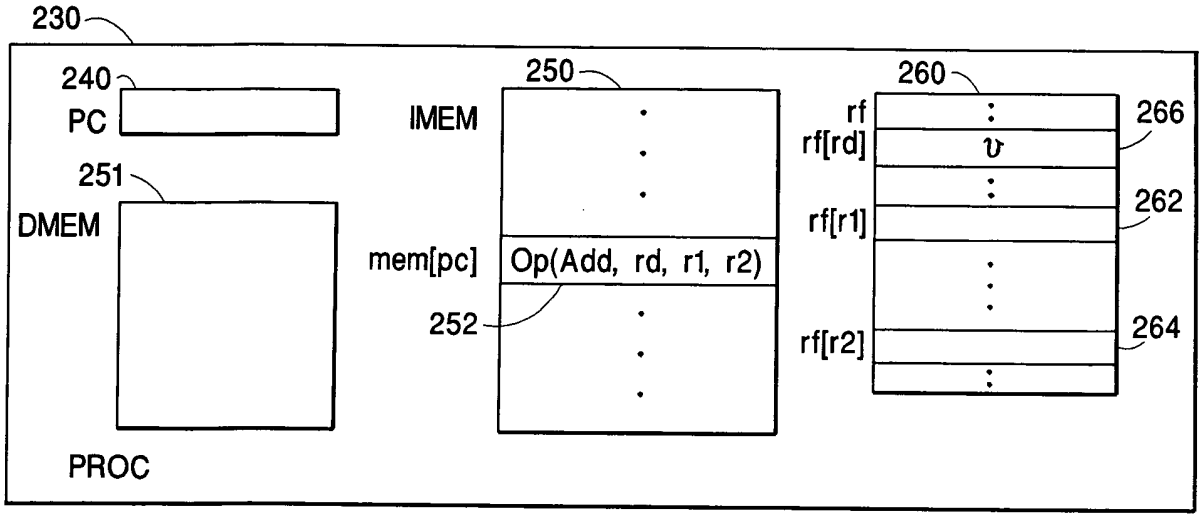
230
240
PC
251
DMEM
PROC
250
IMEM
mem[pc]
Op(Add, rd, r1, r2)
252
260
rf
rf[rd]
v
rf[r1]
rf[r2]
266
262
264

FIG. 2A

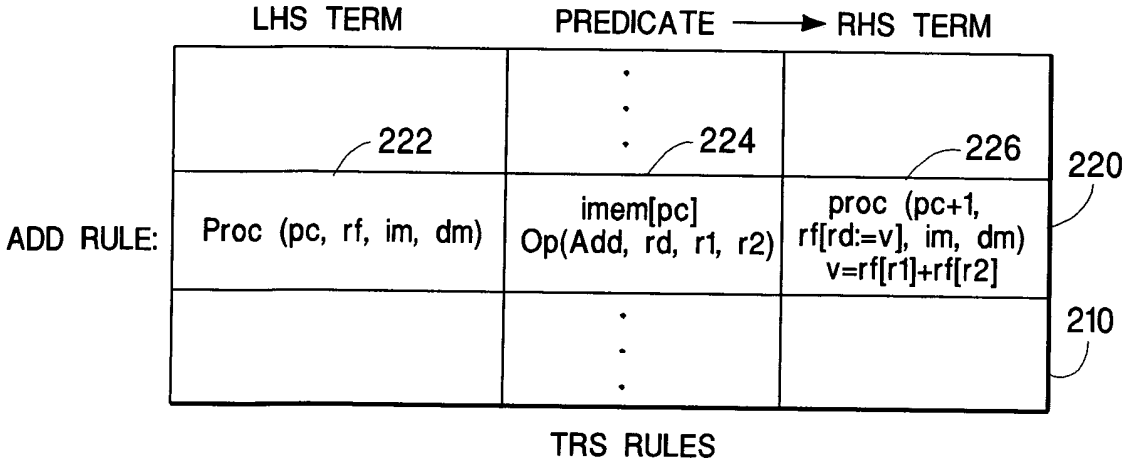
LHS TERM
PREDICATE
RHS TERM
222
224
226
220
ADD RULE:
Proc (pc, rf, im, dm)
imem[pc]
Op(Add, rd, r1, r2)
proc (pc+1,
rf[rd:=v], im, dm)
v=rf[r1]+rf[r2]
210
TRS RULES

FIG. 2B

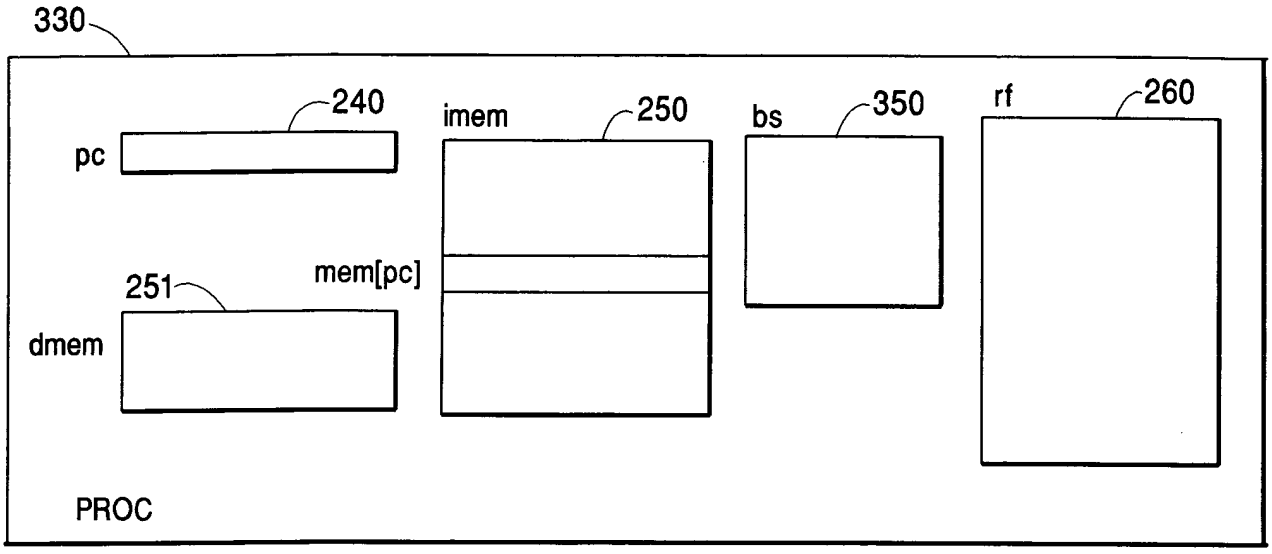
330
240
pc
imem
250
bs
350
rf
260
251
mem[pc]
dmem
PROC

FIG. 3A

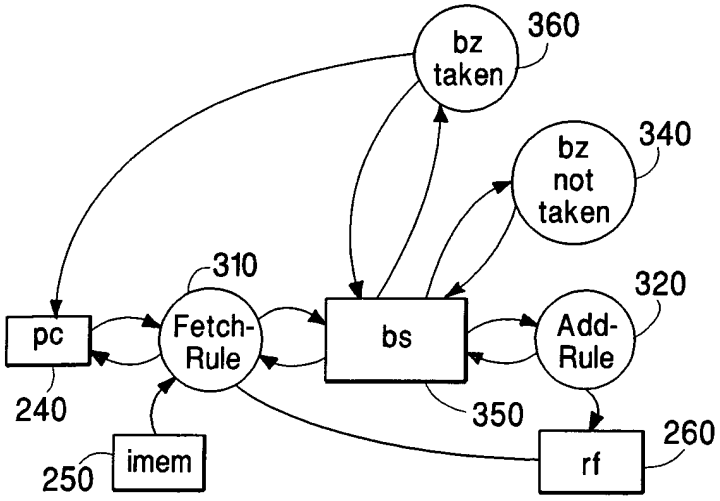
bz taken
360
bz not taken
340
310
320
pc
Fetch-Rule
bs
Add-Rule
240
350
260
250
imem
rf

FIG. 3B

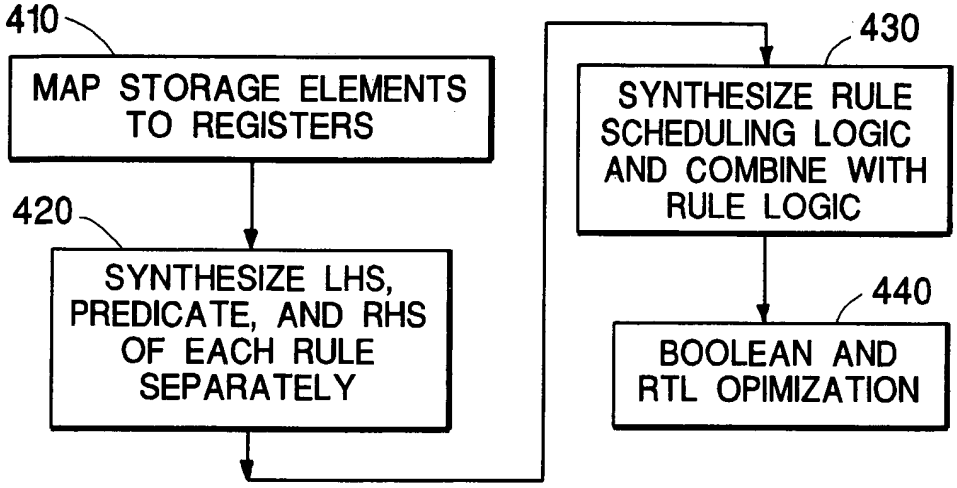
410
MAP STORAGE ELEMENTS TO REGISTERS
420
SYNTHESIZE LHS, PREDICATE, AND RHS OF EACH RULE SEPARATELY
430
SYNTHESIZE RULE SCHEDULING LOGIC AND COMBINE WITH RULE LOGIC
440
BOOLEAN AND RTL OPIMIZATION

FIG. 4

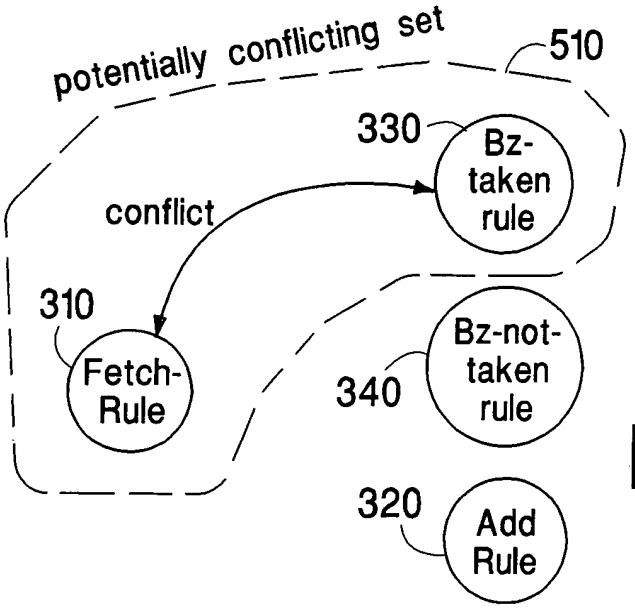
potentially conflicting set
510
330
Bz-taken rule
conflict
310
Fetch-Rule
340
Bz-not-taken rule
320
Add Rule

FIG. 5

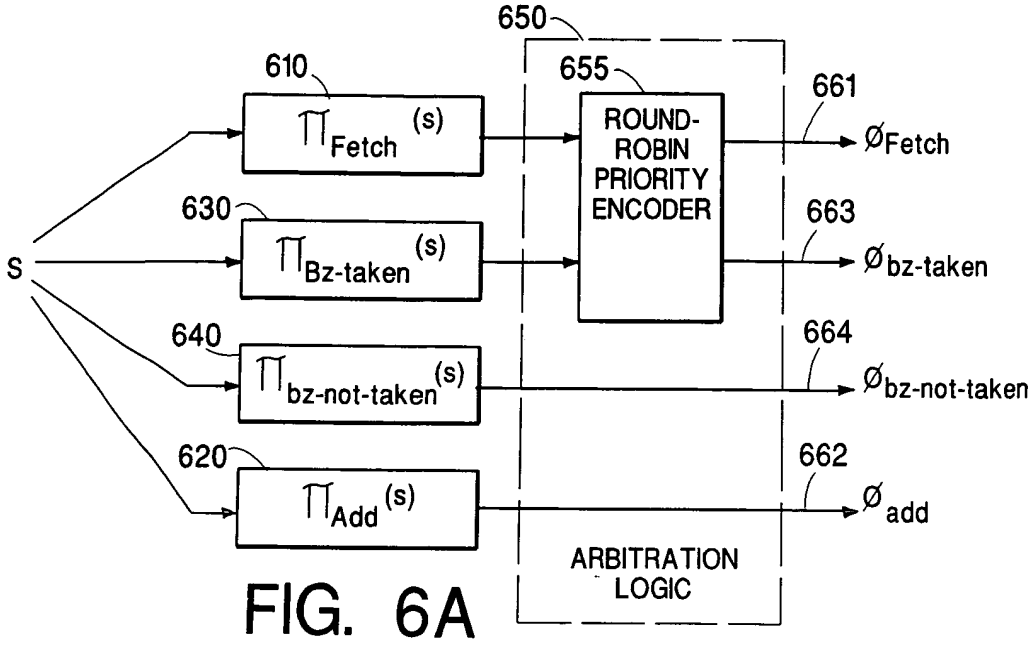

FIG. 6A

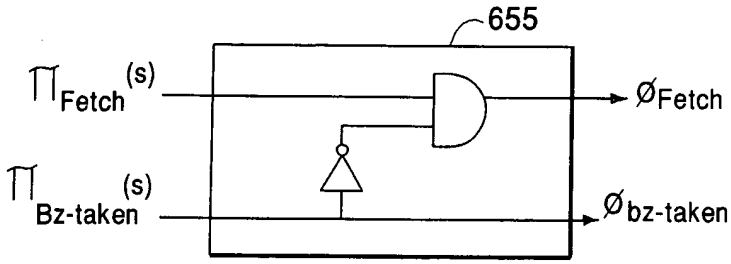

FIG. 6B

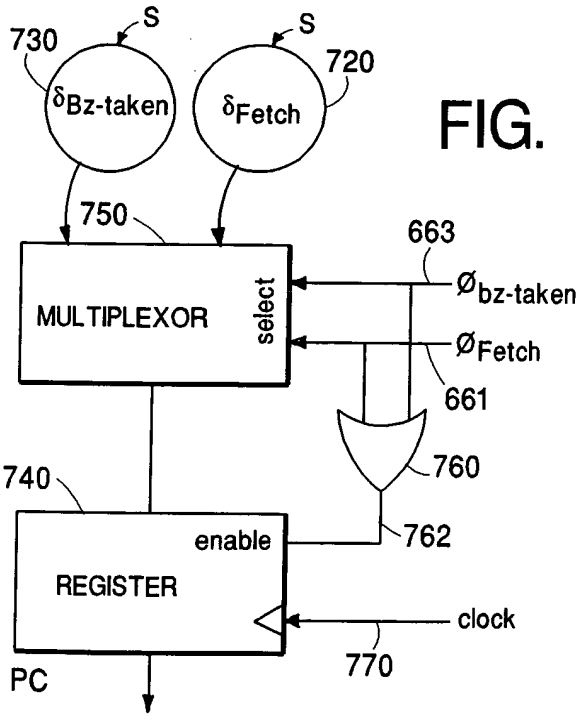

FIG. 6C

```
701 Type    PROC = Proc(PC,RF,IMEM,DMEM)
702 Type      PC = Bit[n]
703 Type    ADDR = Bit[n]
704 Type   VALUE = Bit[n]
705 Type      RF = Array VALUE[RNAME]
706 Type   RNAME = Reg0 || Reg1 || Reg2 || Reg3 || .... Regm
707 Type    IMEM = Array INST[PC]
708 Type    DMEM = Array VALUE[ADDR]
709 Type     INST = Loadc(RNAME,VALUE)
710                || Loadpc(RNAME)
711                || Op(MINOR,RNAME,RNAME,RNAME)
712                || Bz(RNAME,RNAME)
713                || Load(RNAME,RNAME)
714                || Store(RNAME,RNAME)
715 Type   MINOR = Add || Sub || ....
```

FIG. 7

```
801 Proc(pc,rf,im,dm)
802 if im[pc]≡Loadc(rd,const)          →  Proc(pc+1,rf[rd:=const],im,dm)
803 if im[pc]≡Loadpc(rd)               →  Proc(pc+1,rf[rd:=pc],im,dm)
804 if im[pc]≡Op(op,rd,r1,r2)          →  Proc(pc+1,rf[rd:=op(rf[r1],rf[r2])],im,dm)
805 if im[pc]≡Bz(rc,rt) & rf[rc]≡0     →  Proc(rf[rt],rf,im,dm)
806 if im[pc]≡Bz(rc,rt) & rf[rc]≠0     →  Proc(pc+1,rf,im,dm)
807 if im[pc]≡Load(rd,ra)              →  Proc(pc+1,rf[rd:=dm[rf[ra]]],im,dm)
808 if im[pc]≡Store(ra,r)              →  Proc(pc+1,rf,im,dm[rf[ra]:=rf[r]])
```

FIG. 8

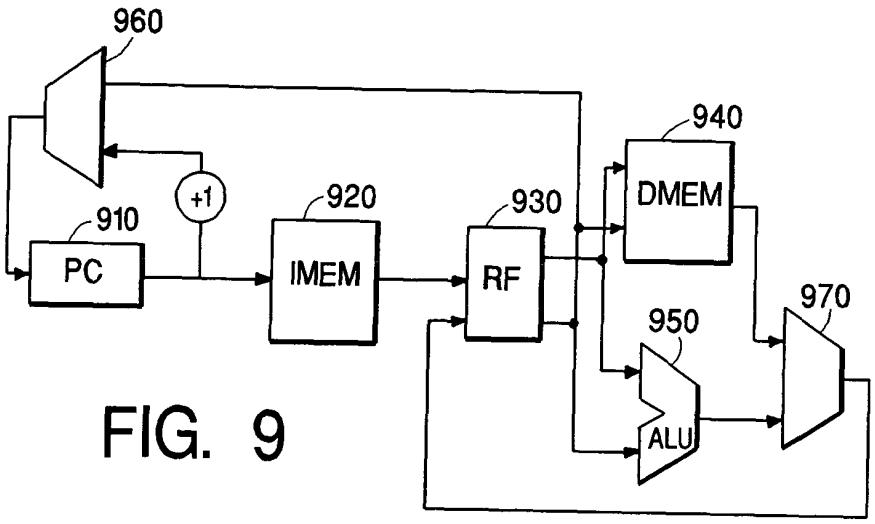
960
940
DMEM
+1
920
930
910
PC
IMEM
RF
970
950
ALU

FIG. 9

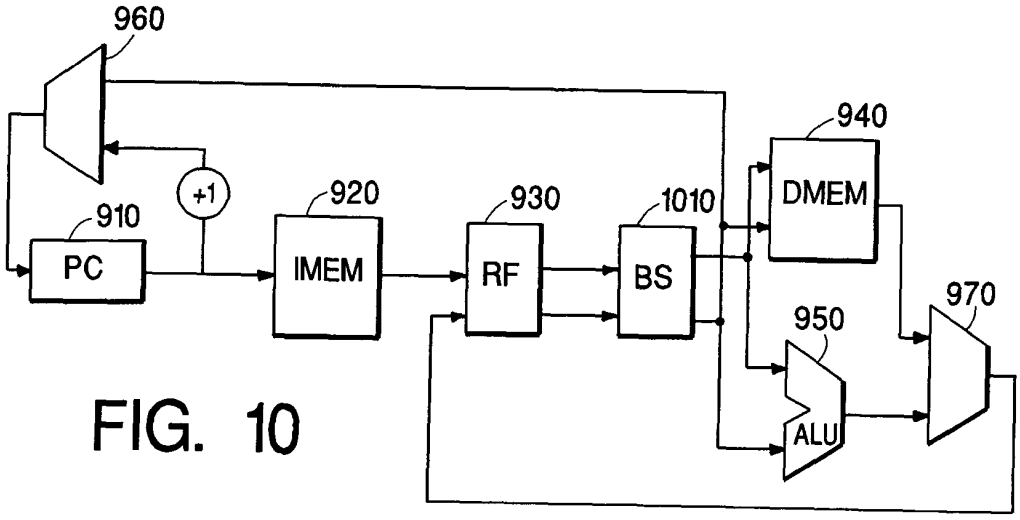
960
940
DMEM
+1
920
930
1010
910
PC
IMEM
RF
BS
970
950
ALU

FIG. 10

```
1101 Type   PROC_p = Proc_p(PC,RF,BS,IMEM,DMEM)
1102 Type       BS = FIFO ITEMP
1103 Type    ITEMP = Op(MINOR,RNAME,VALUE,VALUE)
1104                || Bz(VALUE,VALUE)
1105                || Load(RNAME,ADDR)
1106                || Store(ADDR,VALUE)
```

FIG. 11

```
1201      Proc_p(pc,rf,bs,im,dm)
1202          if  im[pc]≡inst and (Source(inst) ∩ Target(bs) ≡ ∅)
1203 →    Proc_p(pc+1,rf,enq(bs,Decode(inst)),im,dm)
```

FIG. 12A

```
1211 Proc_p(pc,rf,bs,im,dm) where itemp:=first(bs)
1212 if itemp≡Op(op,rd,v1,v2)      →  Proc_p(pc,rf[rd:=op(v1,v2)],deq(bs),im,dm)
1213 if itemp≡Bz(vc,vt) & vc≡0     →  Proc_p(vt,rf,clear(bs),im,dm)
1214 if itemp≡Bz(vc,vt) & vc≠0     →  Proc_p(pc,rf,deq(bs),im,dm)
1215 if itemp≡Load(rd,va)          →  Proc_p(pc,rf[rd:=dm[va]],deq(bs),im,dm)
1216 if itemp≡Store(va,v)          →  Proc_p(pc,rf,deq(bs),im,dm[va:=v])
```

FIG. 12B

```
1301      Proc_p(pc,rf,bs,im,dm)
1302          if  im[pc]≡inst  and  im[pc+1]≡inst'
1303        and   Source(inst)∩Target(bs) ≡ ∅
1304        and   Source(inst')∩(Target(bs)∪Target(inst)) ≡ ∅
1305  →   Proc_n((pc+1)+1,rf,enq(enq(bs,Decode(inst)),Decode(inst')),im,dm)
```

FIG. 13A

```
1311 Proc_p(pc,rf,bs,im,dm) where Op(op,rd,v1,v2):=first(bs), itemp:=first(deq(bs))
1312 if itemp≡Op(op',rd',v1',v2')   →   Proc_p(pc,(rf[rd:=op(v1,v2)])[rd':=op'(v1',v2')],deq(deq(bs)),im,dm)
1313 if itemp≡Bz(vc,vt) & vc≡0     →   Proc_p(vt,rf[rd:=op(v1,v2)],clear(bs),im,dm)
1314 if itemp≡Bz(vc,vt) & vc≠0     →   Proc_p(pc,rf[rd:=op(v1,v2)],deq(deq(bs)),im,dm)
1315 if itemp≡Load(rd',va)          →   Proc_p(pc,(rf[rd:=op(v1,v2)])[rd':=dm[va]],deq(deq(bs)),im,dm)
1316 if itemp≡Store(va,v)           →   Proc_p(pc,rf[rd:=op(v1,v2)],deq(deq(bs)),im,dm[va:=v])
```

FIG. 13B

1321 $Proc_p(pc,rf,bs,im,dm)$ *where* $Bz(vc,vt):=first(bs)$, *itemp*:=$first(deq(bs))$
1322 *if* *itemp*≡$Op(op,rd,v1,v2)$ & $vc\neq0$ → $Proc_p(pc,rf[rd:=\overline{op}(v1,v2)],deq(deq(bs)),im,dm)$
1323 *if* *itemp*≡$Bz(vc',vt')$ & $vc\neq0$ & $vc'\equiv0$ → $Proc_p(vt',rf,clear(\overline{bs}),im,dm)$
1324 *if* *itemp*≡$Bz(vc',vt')$ & $vc\neq0$ & $vc'\neq0$ → $Proc_p(pc,rf,deq(deq(bs)),im,dm)$
1325 *if* *itemp*≡$Load(rd,va)$ & $vc\neq0$ → $Proc_p(pc,rf[rd:=dm[va]],deq(deq(bs)),im,dm)$
1326 *if* *itemp*≡$Store(va,v)$ & $vc\neq0$ → $Proc_p(pc,rf,deq(deq(bs)),im,dm[va:=v])$

FIG. 13C

1331 $Proc_p(pc,rf,bs,im,dm)$ *where* $Load(rd,va):=first(bs)$, *itemp*:=$first(deq(bs))$
1332 *if* *itemp*≡$Op(op,rd',v1,v2)$ → $Proc_p(pc,(rf[rd:=dm[va]])[rd':=\overline{op}(v1,v2)],deq(deq(bs)),im,dm)$
1333 *if* *itemp*≡$Bz(vc,vt)$ & $vc\equiv0$ → $Proc_p(vt,(rf[rd:=dm[va]]),clear(\overline{bs}),im,dm)$
1334 *if* *itemp*≡$Bz(vc,vt)$ & $vc\neq0$ → $Proc_p(pc,(rf[rd:=dm[va]]),deq(deq(bs)),im,dm)$

FIG. 13D

1341 $Proc_p(pc,rf,bs,im,dm)$ *where* $Store(va,v):=first(bs)$, *itemp*:=$first(deq(bs))$
1342 *if* *itemp*≡$Op(op,rd,v1,v2)\}$ → $Proc_p(pc,rf[rd:=\overline{op}(v1,v2)],deq(deq(bs)),im,dm[va:=v]$
1343 *if* *itemp*≡$Bz(vc,vt)$ & $vc\equiv0$ → $Proc_p(vt,rf,clear(\overline{bs}),im,dm[va:=v])$
1344 *if* *itemp*≡$Bz(vc,vt)$ & $vc\neq0$ → $Proc_p(pc,rf,deq(deq(bs)),im,dm[va:=v])$

FIG. 13E